Accessibility in Computing

CS 390 Collaborative Projects – From Concept to Delivery
Why should I care?  
Part 2: Who is disabled?

You are (or you will be at some point in some way).

Accommodations originally made to help with disabilities often benefit everyone.

1.1 Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.

...Computing professionals should consider whether the results of their efforts will...be broadly accessible...

3.7 Recognize and take special care of systems that become integrated into the infrastructure of society.

...When organizations and groups develop systems that become an important part of the infrastructure of society, their leaders have an added responsibility to be good stewards of these systems. Part of that stewardship requires establishing policies for fair system access, including for those who may have been excluded...

acm.org/code-of-ethics
Don’t take it from me
Frameworks and Mindsets

**Accommodations**
Provide a way to request additional help or features when the “default” version does not work well for everyone.

**Universal Design**
“the design of products & environments to be usable by all people, to the greatest extent possible, without the need for adaptation or specialized design.”

The Center for Universal Design
design.ncsu.edu/cud
POUR Accessibility Principles

Computational artifacts should be:

1. Perceivable: by sight, sound, and touch (multiple senses at once).
2. Operable: by selecting, inputting, searching.
3. Understandable: simple, consistent, and predictable (no surprises).
4. Robust: to assistive technologies and evolving uses/users.
WCAG Standards

w3.org/WAI/standards-guidelines/wcag/
## WCAG POUR Standards

<table>
<thead>
<tr>
<th>Principles</th>
<th>Guidelines</th>
<th>Level A</th>
<th>Level AA</th>
<th>Level AAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Perceivable</td>
<td>1.1 Text Alternatives</td>
<td>1.1.1</td>
<td>1.2.1 – 1.2.3</td>
<td>1.2.4 – 1.2.5</td>
</tr>
<tr>
<td></td>
<td>1.2 Time-based Media</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.3 Adaptable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.4 Distinguishable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Operable</td>
<td>2.1 Keyboard Accessible</td>
<td>2.1.1 – 2.1.2</td>
<td>2.1.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.2 Enough Time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.3 Seizures</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.4 Navigable</td>
<td>2.4.1 – 2.4.4</td>
<td>2.4.5 – 2.4.7</td>
<td>2.4.8 – 2.4.10</td>
</tr>
<tr>
<td>3. Understandable</td>
<td>3.1 Readable</td>
<td></td>
<td>3.1.1 – 3.1.6</td>
<td>3.1.2 – 3.1.6</td>
</tr>
<tr>
<td></td>
<td>3.2 Predictable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.3 Input Assistance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Robust</td>
<td>4.1 Compatible</td>
<td>4.1.1 – 4.1.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Low Hearing and Deafness

Synchronous Captions

Asynchronous Transcripts

Tips: For video/audio content, integrate captions and provide transcripts.
Low Vision Disabilities

Macular Degeneration
Glaucoma
Diabetic Retinopathy
Cataracts

Tips: Use high contrast, make text expandable
“Color-Blindness” (es)

Protanopia (red cone deficiency, a.k.a. “red-green color-blind”)

Tips: Never rely on color alone to communicate, never use red/green as a primary contrast
Checking Color Contrasts

Contrast Checker

Contrast Ratio: 1.89:1

Normal Text:
WCAG AA: Fail
WCAG AAA: Fail

The five boxing wizards jump quickly.
Blindness and Screen Readers

Navigate by headers and tabbing, *not* by mouse / GUI.

Tips: Always use correct structured headers (h1, h2, …), provide alt-text for images, and ensure all forms can be completed by tabbing.
## Motor Impairment

**High Level Tips:** Make interactivity workable with a keyboard and assume there will be accidental mis-clicks.

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users may not be able to use the mouse.</td>
<td>Ensure all functions are available to both mouse/touchpad and keyboard users (try Tabbing from link to link).</td>
</tr>
<tr>
<td>Users may not be able to control the mouse or the keyboard well.</td>
<td>Ensure forms are error-tolerant (e.g., ask &quot;are you sure you want to delete this file?&quot;). Create sufficiently large links and buttons that remain in a static position. Allow users to remap or disable single key shortcuts.</td>
</tr>
<tr>
<td>Users may be using voice-activated software.</td>
<td>Voice-activated software can replicate mouse movement, but not as efficiently as keyboard functionality. Ensure all functions are available from the keyboard. Provide descriptive link and button text.</td>
</tr>
<tr>
<td>Users may become fatigued when using adaptive technologies.</td>
<td>Provide a method for skipping over long lists of links or other lengthy content.</td>
</tr>
<tr>
<td>Users may not be able to physically interact with their hardware device</td>
<td>Ensure content works in both horizontal or vertical orientation. Do not rely on motion actuation (such as shaking or panning the device) or pointer gestures (such as swiping or dragging).</td>
</tr>
</tbody>
</table>
Not all fixes are hard and time-consuming

Adding alt text

EXAMPLE:
0123 456 7890

CODE SNIPPET:
<p>
<img src="phone.png" alt="Telephone:"> 0123 456 7890
</p>

Labeling form input

EXAMPLE:
First name:  
<input type="text" name="firstname" id="firstname"><br>
<input type="checkbox" name="subscribe" id="subscribe">
<label for="subscribe">Subscribe to newsletter</label>

CODE SNIPPET: HTML
<label for="firstname">First name:</label>  
<input type="text" name="firstname" id="firstname"><br>
<input type="checkbox" name="subscribe" id="subscribe">
<label for="subscribe">Subscribe to newsletter</label>

Find these and other simple web tutorials at w3.org/WAI/tutorials/
For mobile perspectives, see blog.usablenet.com/mobile-app-accessibility-techniques-for-inclusive-design-part-1
You don’t have to do it all from scratch

Most existing frameworks have tools to help you build and audit for accessibility.

See [flutter.dev/docs/development/accessibility-and-localization/accessibility](https://flutter.dev/docs/development/accessibility-and-localization/accessibility) for example.

Accessibility Activity

• Create a wiki page for an Accessibility Plan

• Two sections:
  • Testing: For each major feature/functionality, list all accessibility questions you can think of to test (examples on next slide).
  • Improvements: Based on your testing, for each major feature/functionality, list concrete tasks that you can take to improve accessibility.

• Add improvement tasks to your backlog.

• As always, consider how much time it will take, who will do it, and by when. Prioritize high value tasks first.
Example Accessibility Questions

1. Can I interact without using a mouse?
2. Are headers labeled as such / is my content marked up structurally?
3. Would I understand all the content without color?
4. Would I understand all the content if it were read to me (without looking)?
5. Am I using enough contrast? Try turning the brightness to minimum – can you still read it?
6. Can I accidentally delete / quit something with a single mis-click?
7. Can the text be resized – either directly or through global settings (e.g., by changing text size on iOS settings)?
8. If I turn the text size up to max, is everything still readable?
9. Do images have descriptive labels / alt text (especially if the image is important like a navigation button)?
10. Do my development frameworks provide accessibility assistance? Are we using them? Are they working?